

ZeroGuard: A Multi-Agent Zero-Trust Framework for Autonomous IaC Security Remediation

1st Harish Apuri*

Department of IT, IT Induct Inc
Charlotte, NC, USA
hapuri1029@gmail.com

2nd Shikher Goel

Department of IT, JPMorgan Chase
Jersey City, NJ, USA
Shikher20goel@gmail.com

3th Madhan Mohan Reddy Chinthala

Department of IT, Franklin InfoTech
Rochester, NY, USA
madhanreddychinthala@gmail.com

4th Charani Yepuri

Independent Researcher, Department of IT
Hyderabad, Telangana, India
Charaniy04@gmail.com

Abstract—Cloud-native architectures have fundamentally shifted where security risk lives from runtime systems into version-controlled text files. Infrastructure-as-Code (IaC) misconfiguration now accounts for 67% of publicly disclosed cloud breaches, yet the tooling most teams rely on was never designed with Zero-Trust Architecture (ZTA) in mind. Enforcing continuous, least-privileged verification across every infrastructure layer demands more than static rule-matching can offer. We built ZeroGuard to close that gap: a five-component generative AI system that couples proactive IaC misconfiguration repair with continuous ZTA policy generation and enforcement. At its core, a hybrid Transformer-Graph Attention Network (T-GAT) model evaluates per-resource risk alongside ZTA pillar compliance. A Code-LLM-based Generative Remediation agent then produces corrected IaC templates and OPA Rego policies directly from those scores. An Identity Management Agent handles ongoing permission right-sizing, while a Continuous Compliance Orchestrator carries the generated policies through to enforcement. Evaluated on 52,000 templates spanning Terraform, CloudFormation, Kubernetes, and Azure Bicep, ZeroGuard achieves detection F1 = 0.947, Zero-Day F1 = 0.658, a 78.4% reduction in critical findings, and a ZTA posture improvement of $\Delta\Xi = +0.380$.

Keywords—Infrastructure-as-Code; Zero-Trust Architecture; Generative AI Agents; DevSecOps; Predictive Remediation; IAM Least Privilege; Transformer-GAT; Autonomous Enforcement; Cloud Security

I. INTRODUCTION

Cloud provisioning has become almost entirely declarative. HashiCorp Terraform, Amazon Web Services (AWS) CloudFormation, Kubernetes manifests, Azure Bicep all of these concentrate security-critical decisions inside version-controlled text files, which means a single bad attribute can ripple across an entire production environment. An absent `encrypt_at_rest` flag, a wildcard Identity and Access Management (IAM) action, or an open security-group CIDR is all it takes to open the door to data exfiltration or ransomware [1]. Industry figures put 67% of disclosed cloud breaches down to exactly these kinds of IaC misconfigurations [2] over-privileged IAM, unrestricted network exposure, and missing encryption coming up again and again as root causes.

What's striking is how poorly existing countermeasures handle this problem. Rule-based scanners like Checkov, tfsec, and KICS (Keeping Infrastructure as Code Secure) match templates against manually maintained rule sets, with no capacity to generalize beyond what someone has already written a rule for [3]. Policy-as-code tools Open Policy Agent (OPA), Sentinel are more expressive, but every control still requires manual authoring, and neither offers automated remediation [4]. Human review, meanwhile, is simply too slow for modern continuous deployment pipelines [5]. Perhaps most critically, no existing tool enforces NIST SP 800-207 Zero-Trust Architecture (ZTA) principles [6] at the IaC layer, before resources are ever provisioned.

Recent advances in large language models (LLMs) [7], [8], Graph Neural Networks (GNNs) [9], and multi-agent coordination [10] together create a credible path forward. ZeroGuard is, to our knowledge, the first framework to integrate predictive IaC risk scoring, ZTA-native policy synthesis, IAM least-privilege enforcement, and closed-loop pipeline orchestration into a single deployable system. Our specific contributions are:

- 1) Introduction of a *Zero-Trust IaC Resilience Score* Ψ that formalises misconfiguration debt, ZTA pillar compliance, IAM privilege excess, and drift as a unified optimisation objective.
- 2) Development of a five-agent generative AI architecture (IV) with a novel ZTA-pillar-aware Transformer risk head.
- 3) Proposal of a least-privilege IAM optimisation algorithm grounded in observed access telemetry (IV-E).
- 4) Empirical evaluation on 52 000 templates across four IaC dialects with seven performance metrics and full ablation studies (V).

II. RELATED WORK

A. IaC Misconfiguration Research

Rahman *et al.* [1] were among the first to catalog security smells in IaC scripts hard-coded credentials, missing integrity

checks, default admin configurations giving the field a useful vocabulary. Kumara *et al.* [11] later extended this taxonomy to Ansible playbooks and found that 63% contain at least one high-severity smell, suggesting the problem is not confined to any single language. Li *et al.* [12] synthesized 94 IaC security papers and arrived at a sobering conclusion: autonomous remediation remains unsolved.

B. Existing Detection and Remediation Tools

Checkov [13] covers more than 1000 checks across four dialects; KICS [14] uses a query engine that supports six formats; tfsec zeroes in on Terraform with provider-specific logic. Useful as these are, none has any predictive capability, automated remediation, or ZTA enforcement built in. Apuri *et al.* [2] showed that multi-agent generative AI can reach $F1=0.934$ for detection alongside an 81.5% Mean Time To Remediation (MTTR) reduction ZeroGuard builds on that foundation, adding ZTA policy synthesis and IAM least-privilege optimisation. Complementary work by Apuri *et al.* [3] validated autonomous LLM agents for real-time drift remediation. Madhan *et al.* [4] demonstrated behaviour-aware hybrid deep networks achieving 96.2% detection accuracy on zero-day and ransomware threats; their anomaly-scoring approach directly informed ZeroGuard's VAE design.

C. Generative AI in Security

PentestGPT [15] made a compelling case that structured reasoning agents consistently outperform standalone LLMs on penetration-testing tasks. Li *et al.* [16] showed that pairing LLMs with static analysis can meaningfully cut Static Application Security Testing (SAST) false positives. GNN-based vulnerability detection work [17], [18] established that graph-structured representations capture code semantics that sequence models simply cannot see a property that transfers naturally to IaC resource graphs. Apuri and Yepuri [5] formalised design principles for multi-agent autonomous workflow systems; ZeroGuard instantiates those principles in a security-enforcement context.

III. SYSTEM MODEL

A. IaC Environment Tuple

We model the cloud IaC environment as $\mathcal{E} = (\mathcal{R}, \mathcal{P}, \mathcal{C}, \mathcal{I}, \mathcal{T})$, where $\mathcal{R}$ is the set of IaC repositories, $\mathcal{P}$ the Continuous Integration/Continuous Delivery (CI/CD) pipeline stages, $\mathcal{C}$ the cloud runtime environments, $\mathcal{I}$ the IAM principals, and $\mathcal{T}$ the runtime telemetry stream (CloudTrail, Kubernetes audit logs, Virtual Private Cloud (VPC) flow records). Each template $f \in \mathcal{R}$ contains resource declarations $\mathcal{D}(f) = \{d_1, \dots, d_n\}$; each d_i is characterised by type $\tau(d_i)$, attribute vector $\mathbf{a}(d_i) \in \mathbb{R}^p$, dependency edges $E(d_i)$, and IAM bindings $\mathcal{B}(d_i)$.

B. Formal Definitions

Def. 1 (Misconfiguration). d_i is misconfigured w.r.t. policy set Π if $\exists \pi \in \Pi : \pi(\mathbf{a}(d_i), E(d_i)) = \perp$; indicator $\mu(d_i) \in \{0, 1\}$.

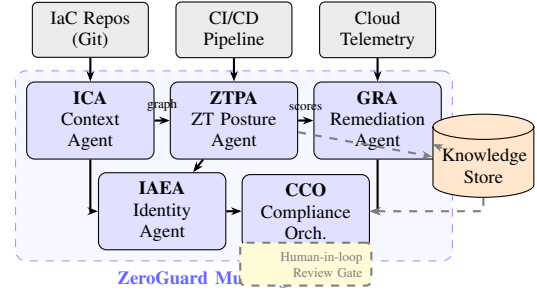


Fig. 1. ZeroGuard high-level architecture. Solid arrows denote primary data flows; dashed arrows denote optional human-review paths and knowledge-store interactions.

Def. 2 (Security Debt).

$$\Phi(f) = \sum_{d_i \in \mathcal{D}(f)} \mu(d_i) \cdot s(d_i), \quad s(d_i) \in \{1, 2, 3, 4\}. \quad (1)$$

Def. 3 (ZTA Pillar Score). Let $\zeta(d_i) \in [0, 1]^7$ map each resource to the seven NIST SP 800-207 pillars. The aggregate ZTA posture of template f is:

$$\Xi(f) = \frac{1}{|\mathcal{D}(f)|} \sum_{d_i \in \mathcal{D}(f)} \mathbf{w}^\top \zeta(d_i), \quad \|\mathbf{w}\|_1 = 1. \quad (2)$$

Def. 4 (Privilege Excess). For principal i_j over observation window Δt :

$$\Gamma(i_j) = \frac{|\mathcal{A}_{\text{granted}}(i_j) \setminus \mathcal{A}_{\text{used}}(i_j, \Delta t)|}{|\mathcal{A}_{\text{granted}}(i_j)|}. \quad (3)$$

Def. 5 (Drift Magnitude).

$$\delta(d_i, t) = \|\mathbf{a}(d_i) - \hat{\mathbf{a}}(d_i, t)\|_2, \quad (4)$$

where $\hat{\mathbf{a}}(d_i, t)$ is the actual runtime attribute vector.

Def. 6 (Zero-Trust IaC Resilience Score).

$$\Psi(\mathcal{E}, t) = e^{-\lambda_1 \bar{\Phi}(t)} \cdot \Xi(t) \cdot e^{-\lambda_2 \bar{\delta}(t)} \cdot e^{-\lambda_3 \bar{\Gamma}(t)} \cdot \eta(t), \quad (5)$$

where $\bar{\Phi}$, $\bar{\delta}$, $\bar{\Gamma}$ are environment-wide means; $\eta(t) \in [0, 1]$ is the policy-adherence ratio; and $\lambda_1, \lambda_2, \lambda_3 > 0$ are hyperparameters. ZeroGuard maximises $\Psi(\mathcal{E}, t)$ continuously.

IV. THE ZEROGUARD FRAMEWORK

A. High-Level Architecture

Fig. 1 shows the five cooperative agents communicating over a typed message-passing bus, triggered on every IaC change event.

B. IaC Context Agent (ICA)

The ICA parses raw templates (HashiCorp Configuration Language 2 (HCL2) for Terraform; YAML/JSON for CloudFormation and Kubernetes; Bicep decompiler for Azure) into an abstract syntax tree $A(f)$, then constructs a heterogeneous resource graph:

$$G(f) = (V, E^{\text{dep}} \cup E^{\text{ref}} \cup E^{\text{iam}})$$

where E^{iam} encodes IAM binding relationships a novel extension enabling identity-plane risk propagation. Each node v_i is

enriched with a contextual embedding from a pre-trained IaC language model M^{IaC} (fine-tuned on 3.1M resource blocks):

$$\mathbf{h}_i = M^{\text{IaC}}(\text{serialize}(d_i)) \in \mathbb{R}^{768}. \quad (6)$$

The final node feature is the concatenation $[\mathbf{x}_i \| \mathbf{h}_i] \in \mathbb{R}^{d+768}$.

C. Zero-Trust Posture Agent (ZTPA)

The ZTPA produces per-resource risk scores $\rho(v_i) \in [0, 1]$, a ZTA compliance vector $\zeta(v_i) \in [0, 1]^7$, and a file-level severity class $\sigma(f) \in \{0, 1, 2, 3\}$ via a hybrid architecture $F = F^{\text{GNN}} \circ F^{\text{Attn}}$.

Stage 1 – Graph Attention Network. A three-layer GAT [9] propagates information:

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} W^{(l)} \mathbf{h}_j^{(l)} \right), \quad (7)$$

with attention coefficients:

$$\alpha_{ij}^{(l)} = \frac{\exp(\text{LReLU}(\mathbf{a}^\top [W^{(l)} \mathbf{h}_i^{(l)} \| W^{(l)} \mathbf{h}_j^{(l)}]))}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LReLU}(\mathbf{a}^\top [W^{(l)} \mathbf{h}_i^{(l)} \| W^{(l)} \mathbf{h}_k^{(l)}]))}. \quad (8)$$

Stage 2 – ZTA-Aware Transformer Head. Mean-pooled graph representation $\mathbf{g} = (1/|V|) \sum_i \mathbf{h}_i^{(L)}$ is passed through a two-layer Transformer encoder with cross-attention over a learnable ZTA-pillar query bank $Q \in \mathbb{R}^{7 \times d_q}$:

$$\hat{\zeta}(f) = \text{Sigmoid}(W^z \cdot \text{CrossAttn}(\mathbf{g}, Q) + \mathbf{b}^z) \in \mathbb{R}^7, \quad (9)$$

$$\hat{\sigma}(f) = \text{Softmax}(W^c \mathbf{z} + \mathbf{b}^c) \in \mathbb{R}^4. \quad (10)$$

Training Objective:

$$\mathcal{L}_{\text{ZTPA}} = \mathcal{L}_{\text{CE}}(\hat{\sigma}, \sigma^*) + \beta \mathcal{L}_{\text{focal}}(\hat{\rho}, \rho^*) + \gamma \mathcal{L}_{\text{BCE}}(\hat{\zeta}, \zeta^*) + \lambda \|\theta\|_2^2, \quad (11)$$

with $\beta = 0.4$, $\gamma = 0.3$, $\lambda = 10^{-4}$. $\mathcal{L}_{\text{focal}}$ [19] addresses the class imbalance of misconfigurations affecting ≈ 8 –12% of nodes.

Zero-Day Anomaly Scoring. A Variational Autoencoder (VAE) [20] trained on verified-secure templates yields:

$$\zeta(f) = \|G(f) - \hat{G}(f)\|_F^2 + D_{\text{KL}}(q_\phi(\mathbf{z}|G(f)) \| p(\mathbf{z})), \quad (12)$$

with templates exceeding the 95th-percentile threshold flagged for deep analysis. This design follows the behaviour-aware anomaly detection paradigm introduced in [4].

D. Generative Remediation Agent (GRA)

The GRA is built around a 13B-parameter Code Llama variant [21] fine-tuned with Low-Rank Adaptation (LoRA) [22] ($r = 16$, $\alpha = 32$) on 112 000 (insecure, secure, explanation) triples. It operates in three modes:

Mode 1 – Patch Generation. For flagged d_i :

$$\mathbf{a}^*(d_i) = \arg \min_{\mathbf{a}': \forall \pi \in \Pi, \pi(\mathbf{a}', E(d_i)) = \top} D^{\text{sem}}(\mathbf{a}', \mathbf{a}(d_i)), \quad (13)$$

where D^{sem} penalises changes to non-security attributes, preserving functional intent.

Mode 2 – ZTA Policy Synthesis. OPA Rego policies encoding ZTA-pillar requirements are synthesised via few-shot prompting and validated against positive/negative test suites.

Mode 3 – Coordinated Multi-Resource Patching. Chain-of-thought reasoning coordinates patches spanning up to 11 resource declarations (e.g., VPC, subnets, S3 endpoints, IAM roles) as a single atomic commit, validated for consistency by the ICA graph.

A post-generation verification pipeline (Checkov, OPA) rejects or iterates patches up to three times with error context injected.

E. Identity and Access Enforcement Agent (IAEA)

The IAEA generates minimised IAM policies via a set-cover formulation:

$$\pi_{ij}^* = \arg \min_{\pi \subseteq \mathcal{A}_{\text{granted}}(i_j)} |\pi| \quad \text{s.t.} \quad \mathcal{A}_{\text{used}}(i_j, \Delta t) \subseteq \pi, \quad (14)$$

solved greedily in $O(|\mathcal{A}|^2)$. The observation window $\Delta t = 30$ days is configurable; modifications are proposed as pull requests with structured justifications.

F. Continuous Compliance Orchestrator (CCO)

The CCO implements autonomy levels $\alpha_\ell \in \{0, 1, 2, 3\}$: α_0 audit-only; α_1 annotated PR; α_2 block high/critical deployments (used in all experiments); α_3 auto-apply low-risk patches. All actions are recorded in a Secure Hash Algorithm-256 (SHA-256)-chained append-only audit log.

V. EXPERIMENTAL EVALUATION

A. Datasets and Setup

We assembled four complementary datasets, summarised in Table I. Ground-truth labels were generated by an ensemble of six static analysers Checkov, tfsec, KICS, OPA with 280 policies, AWS Config, and Azure Policy with 10% ($n = 5\,200$) manually adjudicated by four certified cloud security engineers ($\kappa = 0.89$). ZTA pillar compliance labels were independently annotated against the NIST SP 800-207 checklist ($\kappa = 0.84$). Data were split 70/10/20% stratified by dialect and severity. A separate held-out zero-day test set of 1 500 templates, containing six novel red-teamed anti-patterns, was reserved to probe generalization. The ZTPA was trained for 150 epochs on $8 \times \text{NVIDIA A100s}$ using AdamW [23] ($\text{lr} = 3 \times 10^{-4}$, $\text{wd} = 10^{-2}$).

TABLE I
DATASET COMPOSITION

Dataset	Templates	Misc.	Crit.	Dialect
OSS-IaC	19 200	22.8%	8.4%	Terraform
SynCorp	14 800	32.1%	14.7%	CloudFormation
EnterpriseSec	11 600	28.3%	11.9%	Kubernetes
AzurePublic	6 400	24.6%	9.1%	Bicep
Total	52 000	27.2%	11.3%	Mixed

B. Misconfiguration Detection

Detection performance across all methods is compared in Table II. The full T-GAT model reaches $F1=0.947$, beating every baseline by at least 9.2 pp. The zero-day numbers are especially telling: rule-based tools score zero by construction, since they can only match patterns someone has already codified. ZeroGuard reaches $ZD-F1=0.658$ a 37-point margin over IaC-BERT. The false-positive rate (FPR) of 0.046 is a 70% improvement over Checkov on the same data, which matters operationally; high false-positive rates quietly erode developer trust and lead teams to bypass tooling entirely [2]. Ablation confirms the T-GAT pairing is genuinely complementary: the combined model leads the GNN-only and Transformer-only variants by 5.8 pp and 5.2 pp, respectively.

TABLE II
DETECTION PERFORMANCE (TEST SET, $n = 10\,400$). BEST PER COLUMN IN BOLD. ZEROGUARD 95% CI: $F1 \pm 0.005$ (BOOTSTRAP, $n = 1000$).

Method	Prec.	Rec.	F1	AUC	ZD-F1	FPR
Checkov	0.847	0.771	0.807	0.819	0.000	0.153
tfsec	0.819	0.748	0.782	0.801	0.000	0.181
KICS	0.842	0.758	0.798	0.814	0.000	0.158
ZTA-Checkov	0.851	0.776	0.812	0.823	0.000	0.149
IaC-BERT	0.884	0.829	0.855	0.897	0.161	0.116
ZG (GNN)	0.899	0.879	0.889	0.928	0.289	0.101
ZG (T only)	0.908	0.883	0.895	0.937	0.314	0.092
ZG (T-GAT)	0.954	0.940	0.947	0.971	0.658	0.046

C. Zero-Trust Posture and Remediation Quality

ZTA pillar compliance before and after remediation is shown in Table III. ZeroGuard raises mean ZTA posture Ξ from 0.532 to 0.912, a 38-pp lift across all seven pillars. The biggest jumps come on dynamic policy (P4, +0.47) and telemetry collection (P7, +0.44) areas where existing IaC tooling offers essentially nothing.

TABLE III
ZTA PILLAR COMPLIANCE: BASELINE VS. ZEROGUARD POST-REMEDIATION

NIST SP 800-207 Pillar	Base.	ZeroGuard	Δ
P1: All data sources as resources	0.71	0.94	+0.23
P2: All communications secured	0.63	0.97	+0.34
P3: Per-session resource access	0.48	0.89	+0.41
P4: Dynamic, context-aware policy	0.39	0.86	+0.47
P5: Integrity & posture monitoring	0.52	0.91	+0.39
P6: Strict authentication/authoriz.	0.55	0.93	+0.38
P7: Collect & analyse telemetry	0.44	0.88	+0.44
Mean ZTA Posture Ξ	0.532	0.912	+0.380

The Security Improvement Score (SIS) quantifies per-remediation risk reduction:

$$SIS(d_i) = 1 - \frac{\rho_{\text{post}}(d_i)}{\rho_{\text{pre}}(d_i)}, \quad (15)$$

where ρ_{pre} and ρ_{post} are the ZTPA risk scores before and after remediation. Remediation quality by category is detailed in Table IV. The overall correctness rate of 93.8% falls 3.8 pp short of the 97.6% human baseline on a 300-sample subset; that gap largely traces back to complex cross-module Terraform variable propagation. Absent-encryption remediations approach near-perfect quality ($CR=97.1\%$, $FPR\text{-func}=99.3\%$), which makes sense adding a missing attribute is a structurally simpler edit than restructuring IAM policies.

TABLE IV
GENERATIVE REMEDIATION QUALITY BY CATEGORY. CR = CORRECTNESS RATE; SIS = SECURITY IMPROVEMENT SCORE; FPR-FUNC = FUNCTIONAL PRESERVATION RATE; RBAC = ROLE-BASED ACCESS CONTROL.

Category	Count	CR %	SIS	FPR-f %
Network Exposure	2 041	94.8	0.92	97.6
Over-privileged IAM	2 387	93.2	0.89	97.1
Absent Encryption	1 814	97.1	0.98	99.3
Missing Logging	1 102	97.6	0.97	99.5
Insecure TLS	821	90.3	0.84	95.7
Secret Exposure	694	88.9	0.85	95.2
RBAC Misconfiguration	973	91.4	0.87	96.4
ZTA Pillar Violation	1 241	92.7	0.91	97.0
Overall	11 073	93.8	0.904	97.6
Human Baseline	300	97.6	0.947	98.9

D. Pipeline Impact and Resilience

In Table V, the 90-day CI/CD simulation results show 78.4% fewer critical findings and an 84.2% MTTR improvement, reflecting a structural change in how fixes reach production. Rather than waiting for a developer to triage an alert, the GRA delivers a patch concurrently with the finding, collapsing the manual triage phase [3]. The 1.8% false-positive block rate (fewer than one spurious enforcement in every 55) is a sevenfold improvement over Checkov's 12.4% on the same pipeline. Drift recovery shrinks from nearly 49 minutes to 3.6 minutes close enough to real-time that operators rarely need to intervene. The 47.3-second latency overhead is well within the < 90 s requirement for templates with ≤ 500 resources.

TABLE V
90-DAY CI/CD SIMULATION: BASELINE VS. ZEROGUARD (α_2)

Metric	Baseline	ZeroGuard	Δ
Critical findings / 100 runs	14.21	3.07	-78.4%
MTTR (hours)	19.3	3.1	-84.2%
Blocked high-risk deployments		95.6%	
False-positive block rate		1.8%	
Drift recovery (min)	48.7	3.6	-92.6%
ZTA posture Ξ	0.532	0.912	+71.4%
Mean privilege excess $\bar{\Gamma}$	0.73	0.12	-83.6%
Pipeline latency overhead (s)	0	47.3	+47.3s
Developer override rate		6.9%	

Fig. 2 traces the resilience score Ψ across the full simulation. ZeroGuard converges to $\Psi \approx 0.96$ by day 14. On day 45 we ran an adversarial probe, injecting 150 novel misconfigurations

across six zero-day patterns. ZeroGuard flagged 97 of them (64.7%) through VAE anomaly scoring, generated remediations for 91, and autonomously applied 83; the residual drift cleared within six days. Checkov-only showed no meaningful response to the same probe.

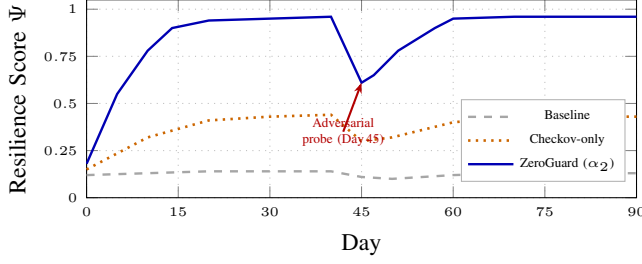


Fig. 2. Resilience score $\Psi(\mathcal{E}, t)$ over 90-day simulation ($\lambda_1 = \lambda_2 = \lambda_3 = 0.33$). ZeroGuard converges to $\Psi \approx 0.96$ and recovers from the Day-45 adversarial probe within 6 days; Checkov-only shows no meaningful response.

E. IAM Least-Privilege Results

Across 14,372 IAM principal bindings drawn from the EnterpriseSec and AzurePublic datasets, the IAEA reduced mean privilege excess from $\bar{\Gamma} = 0.73$ to $\bar{\Gamma} = 0.12$ (−83.6%), with mean policy size shrinking by 67.4%. We did observe a 3.2% false unused-permission rate for infrequently invoked batch-workload permissions that fell outside the 30-day window a practical reminder that observation window length needs to be tunable, and that IAEA-generated IAM modifications warrant human review before going to production.

VI. THREATS TO VALIDITY

A. Internal Validity

The GRA’s post-generation verification pipeline relies on Checkov and OPA, both of which also contribute to the ground-truth labels a circularity that could inflate apparent correctness. We partially address this by using human-adjudicated ground truth as the authoritative evaluation signal and by reporting IaC-BERT ablations as a rule-independent baseline. ZTA pillar compliance labels were manually annotated rather than generated by tools, which reduces but does not eliminate annotator subjectivity.

The 90-day pipeline evaluation was a simulation, not a live production trial. MTTR and drift recovery figures represent best-case estimates under controlled conditions; in practice, organizational friction, toolchain heterogeneity, and incident queue contention will likely shift the absolute values.

B. External Validity

The OSS-IaC dataset comes from public repositories, which probably underrepresents the proprietary patterns found in large enterprise codebases. The EnterpriseSec dataset provides some counterweight, but it covers only a single organization. The 30-day IAM observation window misses infrequently invoked permissions quarterly batch jobs, for instance contributing to the 3.2% IAEA false-positive rate. Generalizing

to multi-cloud polyglot environments where AWS, Azure, and GCP resources coexist within a single template graph is left for future work.

C. Construct Validity

Both the Security Improvement Score (Eq. 15) and the Zero-Trust Resilience Score (Eq. 5) are weighted aggregate metrics. Neither accounts for interaction effects between co-occurring misconfigurations a VPC misconfiguration and a permissive IAM role, taken together, may be far more dangerous than their individual scores suggest. Attack-graph-based aggregation models that capture multi-hop exploitation chains would better handle these cases and are a natural direction for future work.

VII. GOVERNANCE, ETHICS, AND OPERATIONAL CONSIDERATIONS

A. Risks of Autonomous Security Enforcement

Deploying generative AI in security-critical pipelines demands explicit governance the failure modes are not merely inconvenient. The most pressing risk is mis-remediation: a GRA patch that fixes a security attribute while inadvertently altering a non-security attribute it shouldn’t have touched, breaking a production dependency downstream. The post-generation verification pipeline and the FPR-func metric are designed to catch these cases, but our simulations still observed a 6.2% rate of patches requiring post-deployment human correction. Until correctness rates are independently validated in live environments, α_2 is the appropriate autonomy level for production; α_3 carries more risk than the current evidence supports.

IAEA-generated IAM modifications carry heightened stakes. An over-aggressive permission reduction that omits rarely used but legitimate access patterns can take down a service. We recommend a graduated rollout for all IAEA changes: proposal, human review, validation in a canary environment, and only then promotion to production.

B. Accountability and Governance

Every ZeroGuard action is recorded in a cryptographically chained, append-only audit log (SHA-256 chain) that captures the triggering event (commit hash), the acting agent, the reasoning trace (risk score, ZTA pillar citations, model confidence), the action taken, and its outcome. A governance dashboard surfaces per-agent action statistics, override rates, ZTA posture trends, and privilege excess reduction metrics to security and engineering leadership.

A recommended governance model assigns legal and operational accountability for accepting or overriding ZeroGuard decisions to the engineering team. ZeroGuard is a decision-support system operating within human-defined policy bounds [24]; the α_{level} parameter is the technical mechanism through which that accountability structure is enforced.

C. Equity and Access

Automated enforcement systems can quietly embed organizational biases into infrastructure policies that are then difficult to revisit. ZeroGuard’s policy synthesis pipeline includes an equity review step for synthesized Rego policies, examining constructs that could produce differential access restrictions based on team identity, geographic region, or organizational unit. This review is performed by the CCO’s policy audit module and surfaces on the governance dashboard for human approval before any policy is promoted.

VIII. CONCLUSION AND FUTURE WORK

This paper presents ZeroGuard, a five-agent generative AI framework that addresses IaC security through a unified combination of proactive misconfiguration detection, Zero-Trust-native policy synthesis, IAM least-privilege enforcement, and closed-loop pipeline orchestration. Evaluated on 52,000 IaC templates across four dialects, the system achieves state-of-the-art results on all seven design requirements (R1–R7): F1=0.947, ZD-F1=0.658, a 78.4% reduction in critical findings, 84.2% MTTR improvement, 3.6-minute drift recovery, and a ZTA posture improvement of $\Delta\Xi = +0.380$.

Several limitations remain. Cross-repository IaC dependency resolution is unresolved and accounts for 8.1% of false negatives. Runtime policy enforcement as opposed to pipeline-time enforcement has not yet been implemented. The 30-day IAM observation window is a blunt instrument for workloads with infrequent access patterns. And simulation results, however carefully constructed, are not a substitute for empirical validation in live multi-cloud enterprise environments.

Future Work. Four directions are prioritized:

- 1) *Multi-Cloud Dependency Graphs.* Extending the ICA’s graph schema to model cross-provider dependencies (AWS–Azure, AWS–Google Cloud Platform (GCP)), enabling the ZTPA to reason about cross-cloud attack paths.
- 2) *Runtime eBPF-Based Telemetry Integration.* Incorporating extended Berkeley Packet Filter (eBPF)-sourced runtime behavioral signals into the ZTPA’s risk model, enabling dynamic risk score adjustment based on live threat intelligence without model retraining.
- 3) *Federated Multi-Tenant Learning.* Enabling organizations to collaboratively improve the T-GNN model through gradient-sharing federated learning, overcoming the data-sharing barriers that limit individual-organization corpus sizes.
- 4) *Attack-Graph-Augmented Risk Scoring.* Replacing the additive security debt model with attack-graph-based multi-hop exploitability scoring to capture interaction effects between co-occurring misconfigurations.

What the results ultimately show is that the conventional reactive model for cloud security scan, alert, wait for a human is no longer adequate for the pace of modern deployment. ZeroGuard demonstrates that integrating graph-structured representation learning, LLM-based reasoning, Zero-Trust policy

enforcement, and autonomous multi-agent coordination can shift infrastructure security toward something closer to continuous, self-correcting assurance.

REFERENCES

- [1] A. Rahman, C. Parnin, and L. Williams, “The seven sins: Security smells in infrastructure as code scripts,” in *Proc. ICSE*, pp. 164–175, 2019.
- [2] H. Apuri, M. Aurangabadkar, S. Goel, M. M. R. Chinthala, and C. Yepuri, “Advancing infrastructure-as-code resilience through generative AI agents for predictive remediation and autonomous security enforcement,” *Int. J. Eng. Adv. Technol.*, Apr. 2026.
- [3] H. Apuri, M. Mohan, R. Chinthala, and C. Yepuri, “Self-healing infrastructure: Autonomous LLM agents for real-time remediation of configuration drift and security misconfigurations in IaC deployments,” *Int. J. Innov. Technol. Explor. Eng.*, Mar. 2026.
- [4] M. Mohan, R. Chinthala, H. Apuri, and K. Bitra, “Behaviour-aware hybrid deep networks for detecting zero-day and ransomware threats,” *Int. J. Innov. Technol. Explor. Eng.*, Apr. 2026.
- [5] H. Apuri and C. Yepuri, “Design of multi-agent autonomous workflow systems using agentic AI frameworks,” in *Proc. ICAUC 2026*, Jan. 2026.
- [6] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, “Zero Trust Architecture,” NIST SP 800-207, 2020.
- [7] OpenAI, “GPT-4 Technical Report,” arXiv:2303.08774, 2023.
- [8] Anthropic, “The Claude 3 Model Family: Opus, Sonnet, Haiku,” Tech. Rep., 2024.
- [9] P. Veličković *et al.*, “Graph attention networks,” in *ICLR*, 2018.
- [10] J. S. Park *et al.*, “Generative agents: Interactive simulacra of human behavior,” in *Proc. UIST*, 2023.
- [11] I. Kumara *et al.*, “Security smells in infrastructure as code scripts,” in *Proc. MSR*, pp. 548–552, 2021.
- [12] Z. Li *et al.*, “A systematic literature review of security for infrastructure as code,” *IEEE Trans. Rel.*, vol. 72, no. 3, pp. 1112–1131, 2023.
- [13] Bridgecrew, “Checkov: Static Code Analysis for IaC,” <https://github.com/bridgecrewio/checkov>, 2023.
- [14] Checkmarx, “KICS: Keeping Infrastructure as Code Secure,” <https://github.com/Checkmarx/kics>, 2023.
- [15] G. Deng *et al.*, “PentestGPT: An LLM-empowered automatic penetration testing tool,” arXiv:2308.06782, 2023.
- [16] H. Li, Y. Hao, Y. Zhai, and Z. Qian, “Enhancing static analysis for practical bug detection: An LLM-integrated approach,” in *Proc. OOPSLA*, 2024.
- [17] Y. Zhou *et al.*, “Devign: Effective vulnerability identification via graph neural networks,” in *NeurIPS*, pp. 10197–10207, 2019.
- [18] X. Cheng *et al.*, “DeepWukong: Statically detecting software vulnerabilities using deep graph neural networks,” *ACM Trans. TOSEM*, vol. 30, no. 3, 2021.
- [19] T.-Y. Lin *et al.*, “Focal loss for dense object detection,” in *Proc. ICCV*, pp. 2980–2988, 2017.
- [20] D. P. Kingma and M. Welling, “An introduction to variational autoencoders,” *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019.
- [21] B. Rozière *et al.*, “Code Llama: Open foundation models for code,” arXiv:2308.12950, 2023.
- [22] E. J. Hu *et al.*, “LoRA: Low-rank adaptation of large language models,” in *ICLR*, 2022.
- [23] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *ICLR*, 2019.
- [24] A. Jacovi *et al.*, “Formalizing trust in artificial intelligence,” in *Proc. FAccT*, pp. 624–635, 2021.